Computer Networks 181 (2020) 107557

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Online classification of user activities using machine learning on network traffic

Víctor Labayen [a], Eduardo Magaña [b,*], Daniel Morató [b], Mikel Izal [b]

[a] Naudit High Performance Computing and Networking S.L., Madrid, Spain
[b] Department of Electrical, Electronic and Communications Engineering, Public University of Navarre, Pamplona, Spain

A B S T R A C T

The daily deployment of new applications, along with the exponential increase in network traffic, entails a growth in the complexity of network analysis and monitoring. Conversely, the increasing availability and decreasing cost of computational capacity have increased the popularity and usability of machine learning algorithms. In this paper, a system for classifying user activities from network traffic using both supervised and unsupervised learning is proposed. The system uses the behaviour exhibited over the network and classifies the underlying user activity, taking into consideration all of the traffic generated by the user within a given time window. Those windows are characterised with features extracted from the network and transport layer headers in the traffic flows. A three-layer model is proposed to perform the classification task. The first two layers of the model are implemented using K-Means, while the last one uses a Random Forest to obtain the activity labels. An average accuracy of 97.37% is obtained, with values of precision and recall that allow online classification of network traffic for Quality of Service (QoS) and user profiling, outperforming previous proposals.

1. Introduction

Network monitoring and analysis are becoming more challenging due to the growth in traffic demands. There are many new applications, causing network management challenges and resulting in an increase in the complexity of identifying the applications present on the network. This identification is useful for applying QoS policies and detecting intrusions, among other tasks. Each application has its own network requirements; therefore, it is essential to identify these requirements to manage and provide the appropriate resources needed to ensure correct operation.

The traditional approaches for network traffic classification are largely based on the recognition of well-known ports and on DPI (deep packet inspection) [5]. However, there are many applications that use dynamic ports, and many online services are transported over HTTP (Hypertext Transfer Protocol), making it more difficult to recognise those services with a port-based approach. The raise in users' privacy concerns has increased the use of HTTPS (secure HTTP), Virtual Private Networks and encrypted tunnels, making service classification based on DPI unfeasible. Moreover, many governments are promoting laws to prevent internet service providers (ISPs) and other companies from inspecting users' data. Due to these factors, new analysis methods such as those based on machine learning techniques have obtained an increasing popularity among academic researchers and industrial users.

This paper is focused on the characterisation of user network activities in a temporal window using machine learning techniques. We designed a novel analysis architecture using unsupervised techniques, such as K-Means, gaussian mixtures (GM) and BIRCH (balanced iterative reducing and clustering using hierarchies), and supervised techniques such as support vector machines (SVM), random forests (RF) and neural networks (NN), respectively. We present a new approach that differs from the existing methods in the ways in which it pre-processes the input data to the automatic learning algorithms and in the quality of the classification results. Previous works consider all user flows in the classification and make classifications separately, running the risk of losing flow information in different layers of the classification. To solve these problems, an extra layer of classification has been added to achieve aggregation of information from all the user's flows. A time window-based approach is used to accomplish traffic classification that allows its online usage in conjunction with the reconfiguration of QoS, security or accounting policies. The sizes of the time windows are selected by experimental evaluation to obtain fast and accurate results. All this is achieved with a flow level window and three steps in the classification system. The code for the implementation of the system and the dataset utilised are being published along with this paper.

The remainder of this paper is structured as follows: Section 2

* Corresponding Author
E-mail addresses: victor.labayen@naudit.es (V. Labayen), eduardo.magana@unavarra.es (E. Magaña), daniel.morato@unavarra.es (D. Morató), mikel.izal@unavarra.es (M. Izal).

https://doi.org/10.1016/j.comnet.2020.107557
Received 21 February 2020; Received in revised form 3 September 2020; Accepted 9 September 2020
Available online 12 September 2020

describes the user activities considered and the traffic capture methodology. Section 3 presents our approach to performing the traffic classification, including both data preprocessing and classification techniques. Section 4 presents the tuning of the parameters for the classification models. Section 5 presents the evaluation of the system with the captured dataset. Section 6 provides overviews of related work in traffic classification, and Section 7 concludes the paper.

2. User activities and traffic traces

In the design of the classification system, we must define the user network activities we wish to distinguish. Additionally, a full set of network traces is needed to train and test the system. Therefore, a traffic capture methodology that allows us to obtain the network traces with associated user activities (labelling) is proposed.

2.1. Types of user activities

Depending on the scenario, classification per specific application may be needed (for cybersecurity policies); or more general classes can be defined (for QoS or user profiling). We will focus on the latter category, in which a small number of traffic classes is sufficient to characterise the network traffic. In a DiffServ QoS scenario, for example, the number of traffic classes is usually very limited to simplify configuration and management. In scenarios such as user profiling, privacy is very important and identifying specific applications in network traffic should be avoided. Therefore, fewer and more general traffic classes are defined to identify different user profiles. With these requirements in mind, we refer to prior literature [3,15] to find simplified classes such as the following:

- *Interactive*: The interactive class contains traffic from applications that execute real-time interactions in order to provide a suitable user experience. Examples of this type of application include remote file editing in Google Docs, any chatting app or using a remote shell session.
- *Bulk data transfer*: Bulk data transfer activity is comprised of the traffic for those applications that perform transfers of large volume data files over the network, usually making use of a significant percentage of the available network bandwidth. Some applications that exhibit this type of network behaviour are SCP/FTP applications, and direct downloads of large files from web servers like Mediafire or Dropbox, among others.
- *Web browsing*: Web browsing activity includes all of the traffic generated while searching and consuming different web pages, including the download of all multimedia content such as images, ads, javascript libraries, or any other web component.
- *Video playback*: Video playback activity includes traffic from applications that consume video in streaming or pseudo-streaming modes. In both cases, the usage of the network involves bursts of packets that are compensated with the use of buffers. Twitch and YouTube are widely known examples of these applications, and they support streaming and pseudo-streaming, respectively.
- *Idle behaviour*: Idle behaviour activity consists of the background traffic generated by the personal computer when the user is idle. There are many applications that do not require user interaction while making use of the network. Automatic processes, antivirus signature updates, operating system updates, connectivity checks, zeroconf and discovery applications are examples of processes that exhibit this type of behaviour.

These activities have been chosen to cover the most common behaviours exhibited by the network traffic from different applications while using a simple and intuitive set of classes. Moreover, the classification of user activities into this group of classes does not compromise privacy since a specific application has not been inferred.

2.2. Traffic datasets

This section covers the methodology used to capture network traffic and the experimental setup used to obtain the *main dataset*. In addition, we present an *external dataset* provided in [6] that will be used to compare our results in a completely different network scenario.

The goal for obtaining the main dataset is to perform traffic captures in both testing and deployment environments with the minimum number of changes required for moving from one to another. Additionally, due to the extensive time investment in dataset generation, this methodology should be independent of the metrics used and the classification procedures described below. This will allow reuse of the dataset in other scenarios.

The capture process is performed using Tshark [20], a Linux CLI tool, although many other options could be used [14]. The capture is performed using a network probe attached to the router that forwards network traffic from/to the user, as shown in Fig. 1. A SPAN (Switched Port Analyzer) session must be enabled in the router to obtain a copy of all the traffic at the network probe. Once the data are in a pcap file format, the relevant fields are extracted and exported to a CSV (comma-separated values) file because it is a lighter format and is easier to handle. Although Tshark implements this functionality, there are some performance issues with high-rate volume traces and this parsing process is performed using *Scapy* [17], a Python library for handling live network traffic and pcap files. All non-TCP or UDP packets are filtered out, as is each packet with an empty payload, since they are not significant for the classification. The CSV file is composed of a list of packets, with one line per packet. Each packet is characterised by the following features: the timestamp of the packet, a designation of being a TCP or UDP packet, the payload size in bytes, the source and destination IP addresses, and the transport ports involved. Pcap and CSV files are used to simplify the training and testing procedures but, in a real deployment, live data processing would be performed.

When performing a traffic capture to collect a dataset with which to train and evaluate the classification system, the type of user activity must be known. In order to obtain the packet trace and the type of user activity, a single host performs an activity from a known traffic class while the generated packets are being captured using Tshark. This allows us to determine the class with which the trace should be associated. This process could be done with multiple simultaneous users if each user only performs activities from a single class and the labels are associated with the user instead of the whole trace. The labels are required for the training phase of the system but, once trained, the classification system itself will provide the labels.

The main dataset is composed of several traffic traces, each belonging to a certain user and activity. For each traffic class, the following activities are performed: bulk (direct download from Mediafire or a university repository, transfer using SCP/SSH, upload using Dropbox), video (YouTube, Twitch, university online classroom), web (web browsing in blogs and news sites, university Moodle) and interactive (Google Docs, CLI sessions with SSH). The full dataset in both pcap and CSV formats is being published along with this paper and can be found at [4]. Several traces have been captured for each traffic class with different underlying applications and characteristics. The main characteristics of the captured traces are shown in Table 1.

We will also use the dataset provided in [6]. This dataset contains labelled traces for traffic classes different to those considered by us, and particularly to the following: p2p, video, web, bulk, chat, email and voip. Some of these classes contain very few traces and will be discarded for the analysis (the web class contains 3 traces, email only 4 traces and p2p 1 trace, while all other classes have at least 20 traces). The main dataset will be used though the analysis of our proposal and this external dataset will be used for the final validation.

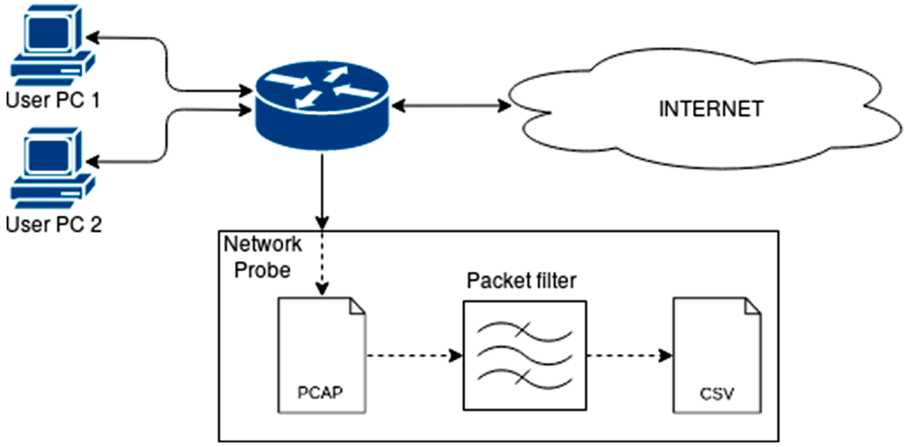

Fig. 1. Capture diagram for the main dataset.

Table 1
Characteristics of data traces in the main dataset.

Activity	File type	Numberof Traces	Total Volume (MB)	Total Duration (s)	Mean Duration (s/trace)	Packets Mean (pkts/trace)	Flows Mean (flows/trace)
Bulk	PCAP	19	8704.13	3598.99	189.42	446738.47	21.53
	CSV		350.86	3586.16	188.75	308832.47	15.53
Video	PCAP	23	1406.34	4494.81	195.43	67577.78	37.04
	CSV		58.35	4458.98	193.87	41691.74	33.52
Web	PCAP	23	148.62	4203.46	182.76	10620.39	240.17
	CSV		8.50	4113.04	178.83	6227.35	235.96
Interactive	PCAP	42	30.58	8934.00	212.71	3141.64	7.00
	CSV		4.46	8932.84	212.69	1783.38	6.98
Idle	PCAP	52	0.69	6341.57	121.95	66.38	7.04
	CSV		0.11	5936.64	114.17	34.85	6.81

3. Classification of user activities

The proposed classification system takes as input a stream of packets with fields from the IP and TCP/UDP headers, as described in Section 2.2. A data preprocessing step is required to transform the network traffic into a set of features that the classification system can handle. The packets are restructured into a three-level hierarchical window data structure that splits the data by user and into temporal windows, making them independent of each other. Then, a three-layer classification process is performed to obtain the user activity associated with each classification window. Fig. 2 shows the workflow of both phases.

3.1. Data preprocessing

Data preprocessing (Fig. 3) is required to describe the network traffic with a set of representative features. First, if the data are provided as multiples traces with a single-user and activity per trace, they are merged into one stream of packets. However, if the capture was made in a network with multiple users, the merging step is not required. If the merging is necessary, all traces are translated to a common starting time, allowing captures made on different days to be merged. The output of this step is a single stream of packets resulting from several users, and hence, the subsequent steps of the process can be independent of the data input format.

The data are split by user (identified by IP address) and by temporal windows (traffic restructuration in Fig. 3), making each window and user independent from the rest. A window filtering operation is performed to remove from the dataset any window with no significant amount of traffic. Finally, feature extraction is performed using the process described in the following sections.

3.1.1. Hierarchical windows

We cannot assume that a user will only perform one single activity during the entire trace, therefore the traffic must be divided into time intervals for classification. These intervals must be chosen so as to capture enough data to constitute a representative depiction, but they must not be so large as to mix several activities. Accordingly, starting with any single-user traffic stream, the traffic will be separated into classification windows. A classification window is the element that the system attempts to classify with one of the activities described in Section 2.1 and is hereinafter referred to as a *CW (Classification Window)*.

Each of these windows can be composed of multiple flows, wherein a flow is defined as the bidirectional conversation of all the packets with the same values in the source and destination IP addresses, source and destination TCP/UDP ports, and protocol field in the IP header. During any activity, the flows present in a CW can have a different degree of relevance in the final classification. In order to allow the classification system to keep that information, the traffic of each of the CWs described earlier is also split into a set of *FW (Flow Windows)*. The FW covers the network traffic for each flow over the duration of the CW. Each CW will

Fig. 2. Diagram for classification phases and workflow.

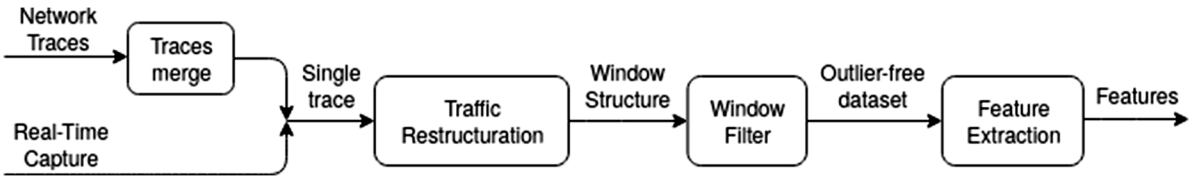

Fig. 3. Preprocessing phases.

have as many FW as the number of flows present in that CW.

Finally, each FW is also split into shorter temporal windows, allowing the characterisation of each FW based on the behaviour exhibited in this set of new windows. This final division is made to allow the system to differentiate between flows containing bursts in traffic, a constant send or receive rate, or those with just one request-response transaction. In the remainder of this paper, we will refer to each of those shorter windows as *BW (Behaviour Windows)*.

The three types of windows are shown in Fig. 4 for the traffic profile of a certain user. In this figure, CW1 has 3 flows and CW2 has only 2 flows. Features of each flow are collected in several BW, but some of them (marked in white in the figure) will be empty as the real duration of the flow can be less than the CW. In the description provided in Section 4, the lengths of CWs and BWs were chosen as 5 s and 0.5 s, respectively, to obtain the best classification results. For such window sizes, the number of packets and flows per window in the main dataset are those shown in Table 2.

3.1.2. Windows filter

A filtering step is performed to remove periods of time not representative of user activity. Several CWs are captured for every user activity exhibiting little traffic, (if any), due to pauses in the user's interaction. The filtering is performed based on the amount of traffic in the CW by selecting a bytes-per-window threshold. As noted earlier, a 5 s CW length is assumed during the traffic restructuring (the length selection is discussed in Section 4). A plot of CW sizes (in bytes) for each traffic activity in the main dataset is shown in Fig. 5. Based on these data, a 3 Kbyte threshold is selected; this is easily exceeded by any active application. Given that web activity has a nearly flat distribution, interactive activity is used to select the threshold such that the majority of samples are not filtered out. In addition, video activity includes a small group of samples exhibiting activity levels at approximately 1 Kbyte/window, and these should be filtered.

Table 3 shows the number of windows from our traffic traces for each activity and type of window, before and after application of the bytes threshold used to filter out low volume windows. We can see that filtered CWs are not from either bulk or idle activities, because bulk has no idle intervals and idle traffic is not being filtered. Meanwhile, several CWs with both video and web activities are filtered because both traffic behaviours have inherent idle intervals. However, the number of FWs and BWs in those activities are reduced by a significantly lower percentage. This is because the CWs that are filtered out contain almost no traffic, and so nothing flows within them.

Table 2
Numbers of packets and flows per hierarchical window in the main dataset.

	Mean	Standard deviation
Packets/CW	1877.04	4381.05
Packets/FW	478.01	2263.63
Packets/BW	47.80	247.53
Flows/CW	3.93	14.12

3.1.3. Feature extraction

Using the CW and the underlying structure, a set of features are defined for each of the window levels. This section describes the features and methodology used to extract windows (BW to CW) from the stream of packets using only statistics determined from the counts, sizes, and inter-arrival times of those packets.

The features used are shown in Table 4. These features are found to be the best choices among all those tested for each window (not listed here for brevity), and they provide optimum results for the full set of metrics initially tested for each level. A feature labelled as "bidirectional" in the table indicates that a feature is obtained for each direction in the communication. Otherwise, only one feature is extracted per sender and receiver pair.

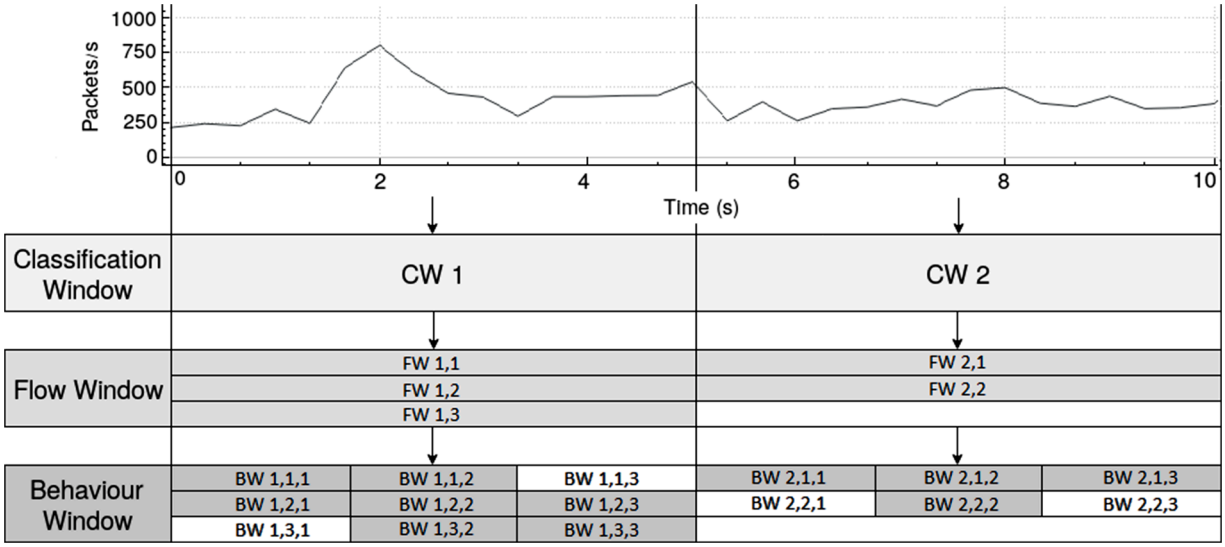

Fig. 4. Hierarchical windows data structure.

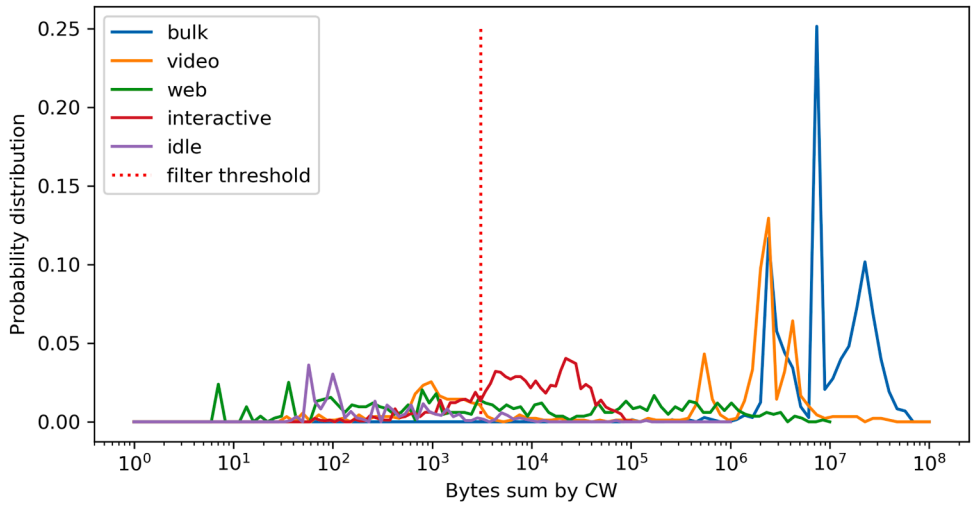

Fig. 5. Probability distribution of bytes per CW and traffic class in the main dataset.

Table 3
Numbers of windows before and after performing window filtering in the main dataset.

Activity	CW count before filter	FW count before filter	BW count before filter	CW count after filter	FW count after filter	BW count after filter
Bulk	728	1120	11200	728	1120	11200
Video	904	1782	17820	493	1544	15440
Web	837	9102	91020	283	8235	82350
Interactive	1809	2211	22110	1173	1860	18600
Idle	1219	526	5260	1219	526	5260
TOTAL	5497	14741	147410	3896	13285	132850

Table 4
Window selection features.

WINDOW	FEATURE	BIDIRECTIONAL
Behaviour Window	Packet Count	No
	Packet Count Send-Receive Ratio	Yes
	Packet Inter-arrival Time Mean	No
	Packet Size Sum	No
	Packet Size Sum Send-Receive Ratio	Yes
	Packet Size Mean	No
	Packet Size Standard Deviation	No
	Packet Size Range	No
Flow Window	Packet Count	No
	Packet Inter-arrival Time Standard Deviation	No
	Packet Size Standard Deviation	No
Classification Window	Packet Count	No
	Packet Size Sum	No
	Flows Count	Yes
	Hosts Count	Yes

3.2. Classification system

Using the hierarchical structure of the three window types, the system will perform a three-layer classification; it proceeds from the first (BW) to the last (CW) and will attempt to infer the prevailing user activity. This means that both the CW and the classification process are completely independent between different windows, and from one user to another. The traffic activities are associated with the CW (third layer), and labels cannot be spread to other layers because different activities could have similar flows and types of behaviours inside each FW. The classification system uses an unsupervised clustering algorithm for the first two layers, assigning clusters to each BW and FW, respectively, to produce a dimensionality reduction in the feature space. The third layer is capable of performing a supervised learning classification. Fig. 6 shows two diagrams for the full process, particularizing for the case of CW2 in Fig. 4.

The first layer (Fig. 6.a) is composed of three steps: feature extraction, window filtering and feature standardisation. Starting from a BW, the features described in Table 4 for a BW are obtained. In the case of an empty window, the BW is ignored in this layer. Each feature is standardised using the mean and standard deviation shown by the same feature among all the BWs of the training set. The standardisation consists of subtracting the mean value and dividing by the standard deviation so that the distribution of that feature presents a mean value of 0 and a standard deviation of 1. This is performed to scale all the features to a similar range so that, regardless of the original magnitude, they are comparable with each other. Finally, using an unsupervised clustering model, each BW is associated with one of the K possible clusters indicating the type of network behaviour exhibited in that temporal interval. The K clusters are obtained from all observations of the traffic for each BW in the training phase.

The workflow of the second layer (Fig. 6.a) is similar to the first layer. Each BW is associated with a cluster while using a special cluster identification for the empty windows. The clusters from all the BWs in an FW are binary-encoded, indicating in the output whether or not a cluster was present in that FW. This binary array is merged with the extracted features from the FW (Table 4), which are standardised with the same methodology used in the first layer. Finally, all the features are applied to a new unsupervised clustering model that will output one cluster identifier for each FW present in the given CW.

Finally, in the third layer (Fig. 6.b), a supervised learning model is used and trained with the activity labels associated with each of the CWs of the training set. The same encoding process is used to obtain a set of features that are representative of the FWs that comprise the CW, and it also merges them with the set of features extracted from the CW (Table 4). That feature set is then normalised as previously described and applied to the supervised learning model. As a result of this process, the identification of the traffic activity is obtained for each CW.

The unsupervised clustering models considered for layers 1 and 2 are K-Means, GM (gaussian mixtures) and BIRCH (balanced iterative reducing and clustering using hierarchies). The supervised learning models considered for layer 3 are SVM (support vector machine), RF (random forest) and NN (neural networks). A comparison of the performance of these methods at different layers is described in the next section.

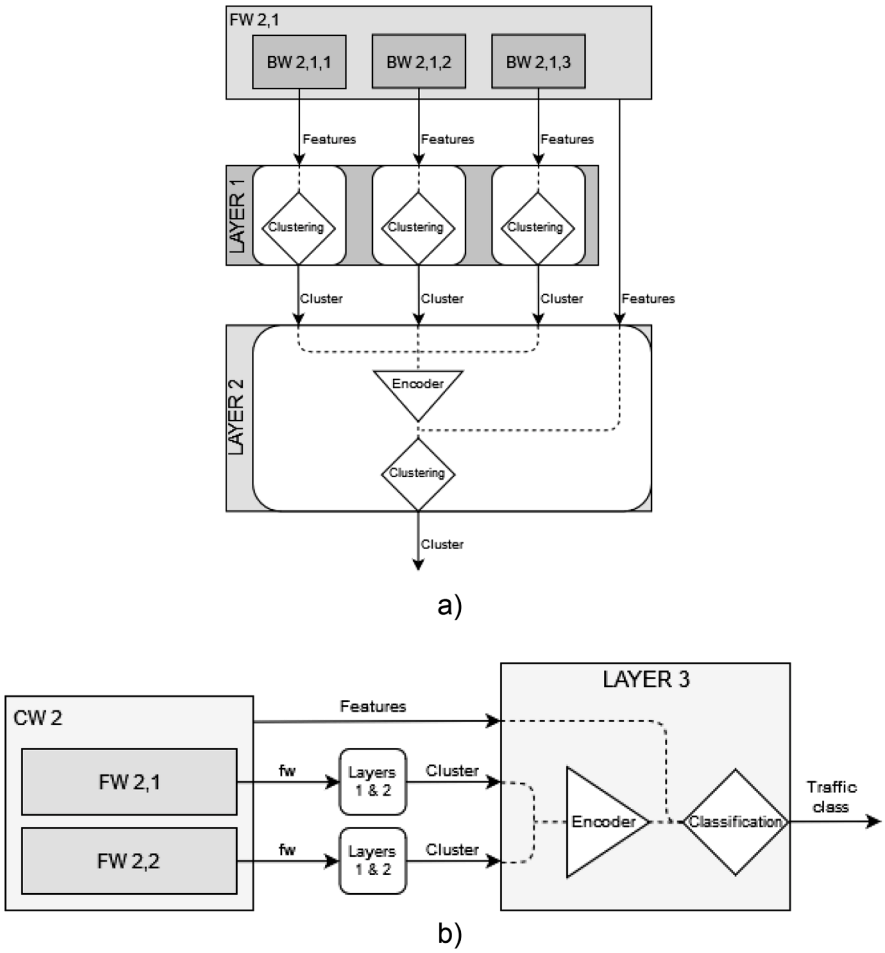

Fig. 6. Three-layer classification system: a) Layers 1 and 2 for a single FW, b) Third layer for a single CW.

4. Tuning the operational parameters

In this section, the results obtained with the classification system are reported, along with the methodology used in the experiments. The classification system is implemented using the Scikit-learn API [13] and the code used to perform the evaluation of the experiments can be found in [4]. The experiments consist of four distinguishable parts: a) performing the data restructuration with the dataset, b) randomly splitting the structured dataset into training and testing sets to perform cross-validation, c) building and training the models, and d) testing the system. The splitting, training and testing processes are repeated up to 100 times to ensure consistency in the results. Each repetition of this sequence will hereinafter be referred to as an iteration. The results are measured using the precision, recall (also called sensitivity) and F1 score metrics. These metrics are defined in expressions (1), (2) and (3), and they are applied to any particular traffic class. TP is the number of True Positive instances (those that are correctly predicted), FP is the number of False Positives (those that are incorrectly assigned to a class), and FN is the number of False Negatives (those that are incorrectly assigned to another class). The F1 score is defined in terms of precision and recall, and is useful for providing a metric that involves both of them. All these metrics assume values ranging from 0 to 1.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1Score = 2\ \frac{Precision\quad Recall}{Precision + \ Recall} \tag{3}$$

To apply our proposed method, we must first define the time lengths for BWs and CWs. Using the main dataset described in Section 2, a parameter-tuning process is performed to select the best combination for those values. This tuning is performed using K-Means as unsupervised clustering at layers 1 and 2 and RF as the supervised classification at layer 3. In Fig. 7 the mean of the classes' recall and precision values are shown for different combinations of lengths for the CW and BW. The results obtained are slightly improved when the lengths of the BWs are greater than 0.3 s. At first glance, any CW value greater than 5 s appears to maximise recall and precision. In addition, CW has to be a multiple of BW. A BW of 0.5 s is selected as a suitable value.

To select the best value for CW, computational complexity and memory usage are also considered. Fig. 8 shows the computational time in seconds and the memory usage in MBytes for the training and testing (without feature extraction) of the full main dataset, with a fixed BW length of 0.5 s and variable CW length (x-axis of the figure). This shows that memory usage is linearly dependent on the CW length, due to the cost of generating features for all the BWs present in the flows of the CW. Conversely, the computational complexity increases with CW values smaller than 5 s, due to the increase in the number of observations for the unsupervised classification model at layer 3.

Memory usage increases with the CW length, hence a CW length of 5 s is chosen to optimise time and memory consumption, and, at the same time, it provides us with enough user traffic to allow the classification of activities. Performing the classification of traffic in windows of 5 s provides enough granularity to be applied in a QoS scenario and it is in

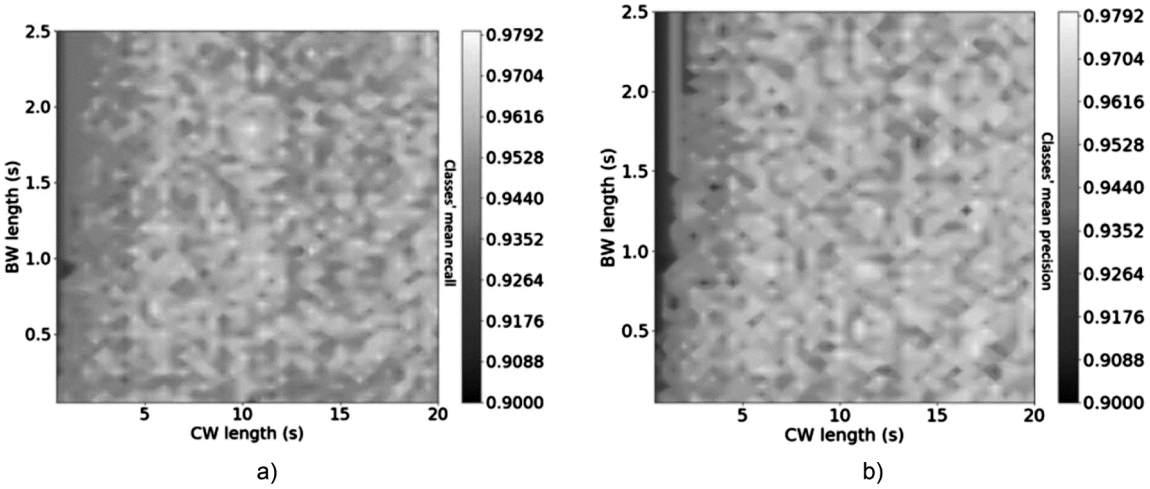

Fig. 7. a) Global recall and b) precision results for CW and BW time lengths (s).

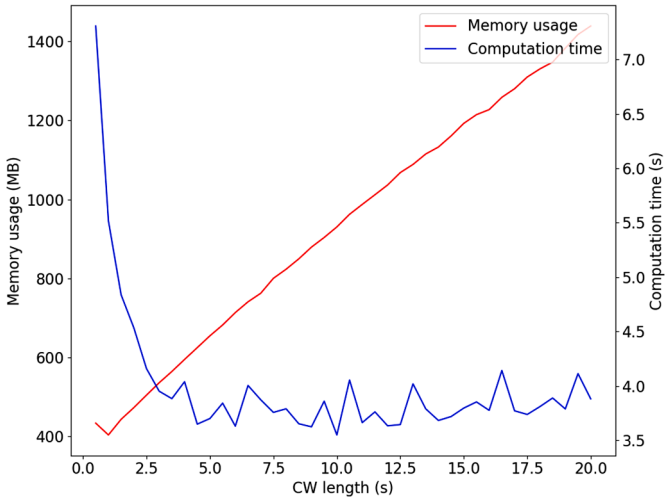

Fig. 8. Computational time and memory usage versus CW length.

the order of magnitude of the time periods when users typically maintain their network activity [19]. Therefore, in what follows, we utilise values of CW=5 s and BW=0.5 s.

The main dataset is split into training and testing sets. In our evaluation the dataset is randomly split by trace (and by user after the data merging process), assigning each trace to the training set with a probability of 60%. The rest of the traces will be used for the evaluation. The models are trained with the training set and then the performance is evaluated using the testing set. All features from Table 4 are considered in the feature extraction process. Fig. 9 shows the average recall and precision for the different classes, resulting from different combinations of the number of clusters in the first and second layer (both axes). The figure shows that 10 or more clusters constitute a suitable number of clusters for the first layer; this is also generally true for the second layer but depends on the number of clusters selected in the first layer. We must also consider that, when the number of clusters in the first layer is too high relative to the number in the second one, the classification performance of the system decays due to an unbalanced overcomplexity. Nevertheless, the second layer performs well with a high number of clusters when the amount in the first layer remains low. This is because there are more possible FW types than there are BW types, as they represent a higher level of complexity.

To select the best combination of FW and BW clusters, it is also important to consider computational time. Fig. 10 shows the time in seconds required to perform both the training and testing steps with the entire main dataset, without including feature extraction. The computational complexity of K-Means is linearly dependent on the number of clusters. In our case, there are always more samples in the first layer than there are in the second one (by a factor equalling the ratio CW-length / BW-length), therefore the number of clusters in the first layer has more impact on computational performance. This is shown in Fig. 9, which

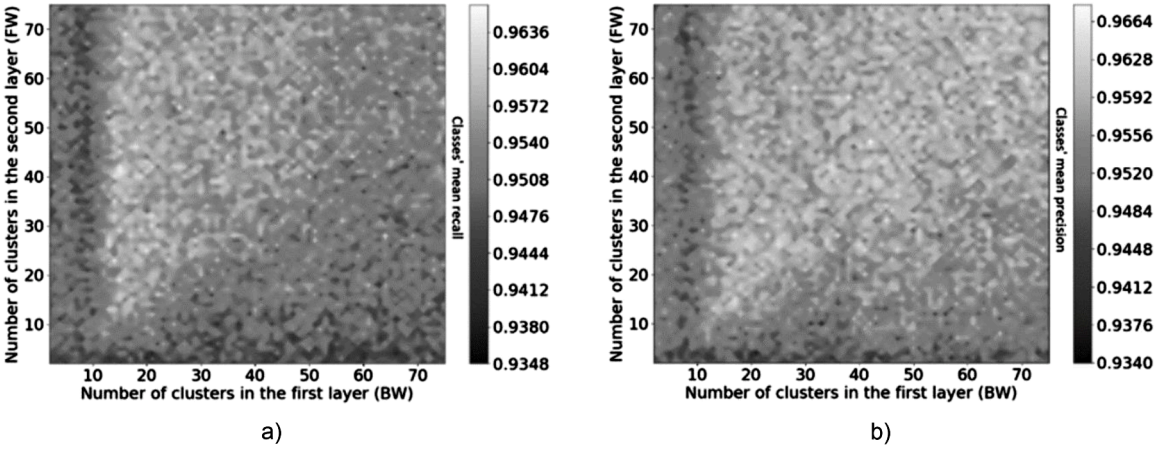

Fig. 9. a) Global recall and b) precision results versus number of clusters.

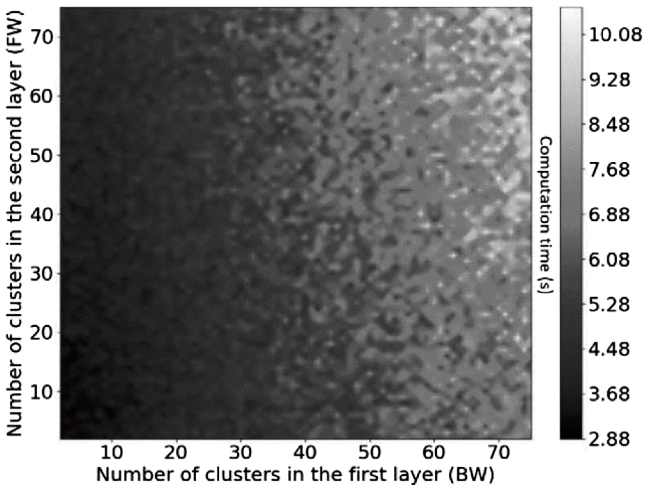

Fig. 10. Computational time (in seconds) versus number of clusters.

represents the vertical (second layer) and horizontal (first layer) slopes. Using the results in Figs. 9 and 10, we select the lowest values for the numbers of clusters in the first and second layer that still maintain high recall values (over 93.7%). These values are 15 clusters for the first layer and 20 clusters for the second layer.

From Fig. 10, a total computational time of 2.9 s is obtained for the selected number of clusters. This computational time is for testing and training the full main dataset. If we consider the sum of the feature extraction time and the computational time for processing a single CW, it is in the order of milliseconds. Therefore, the proposal is able to classify each CW (5 s) almost immediately after its network traffic has been captured, allowing an online operation.

4.1. Online performance

We have selected the operational parameters to obtain the best computational times, however, they could still result in processing times incompatible with online operation over real traffic. To allow this online operation we have to consider the feature extraction time from network traffic and the time to perform the classification for each CW. The sum of both times have to be equal or less that the CW (5 s) to allow online operation.

The feature extraction time is the time needed to obtain the metrics shown in Table 4 from network traffic. We have measured the time needed to parse the data in the main dataset and perform feature extraction for each CW, in what is shown as the restructure time in Fig. 11. We used a low end CPU (Intel Core i5-4570 3.20GHz) and custom python software to obtain these results. The restructure time has a linear dependence on the number of packets per CW. In the 5 s interval of the CW, over 750,000 packets can be processed, which translates into over 9 Gbps with Ethernet full-sized packets. This isa higher rate than needed for a typical user. More importantly, this task can be performed in parallel to packet capture in real network probes. Nowadays, network probes are able to capture, analyse and store 10 Gbps and higher data rates with off-the-shelf systems [12]. The metrics needed in our classification system are very simple compared to the complex metrics that these network traffic analysis systems are able to provide for application performance monitoring. Most of the computation time in our software was dedicated to moving packets to memory and dissecting packet headers. These tasks are already carried out by any traffic analysis probe, therefore, the computation of our metrics is a minimal overhead. Our metrics can be obtained in real time in any of those systems. This means that each CW we would have all the metrics available to perform classification right after the last packet in the CW under analysis, without noticeable delay.

The online operation of our system is not affected by the feature extraction time, but it could be limited by the classification time, which can only take place after the CW time is completed. We need a classification time below the 5 s CW time to avoid an unstable system where more classification work arrives that it can be carried out. Fig. 12 shows the time needed to perform the classification of a single CW, depending on the number of flows present. It shows the contribution from each classification layer. From the figure,the classification time of a single CW from a user with a small number of flows is 50ms and for 100 concurrent flows it is close to 55ms. The average number of concurrent flows per user in the main dataset is 3.93 flows/CW, therefore 100 flows/CW is a worst-case configuration and the computational time gets incremented slowly with the number of concurrent flows.

Fig. 13 shows the classification time for populations of simultaneous users with an average of 3.93 flows/CW (the profile found in the main dataset). The classification time suffers a close to linear increase for populations above 20 users. Below 20 users, the added overhead from the function calls to the machine learning library is noticeable. For larger populations, the processing time remains below 1 s, up to 2,500 simultaneous users. Even using a low end CPU, several gigabits per second of traffic from thousands of users can be classified without more than one or two seconds time-delay. We believe this response time is enough for online operation in the use-cases described in this paper.

5. Classification results

After choosing the operational parameters, we evaluated the system for different classification systems in each layer using the testing traces of the main dataset. The unsupervised clustering models at layers 1 and

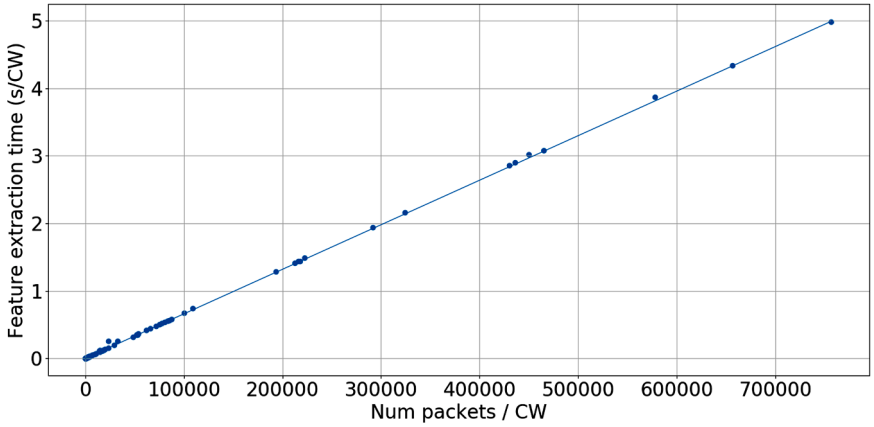

Fig. 11. Restructure time (feature extraction) versus number of packets per CW.

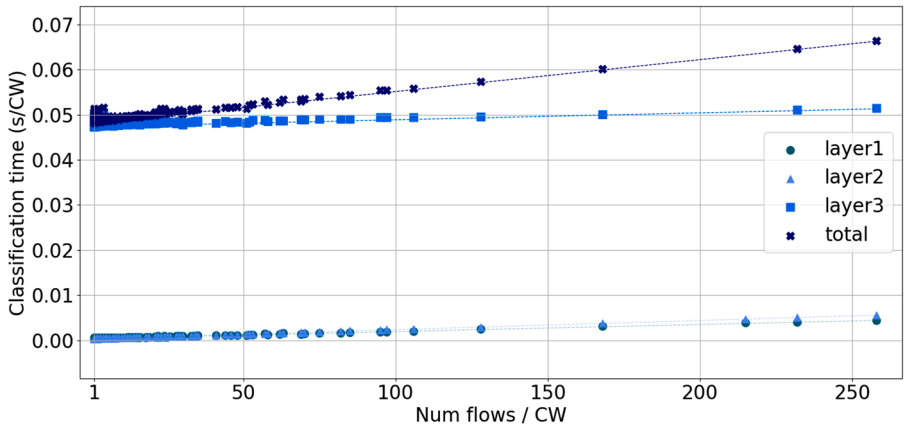

Fig. 12. Classification time versus number of flows per CW.

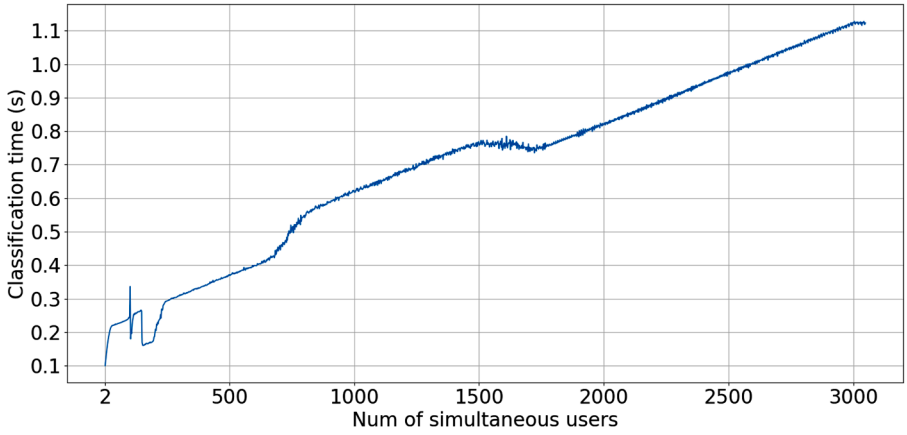

Fig. 13. Classification time versus number of simultaneous users (with 3.93 flows/CW each)

2 are K-Means, GM and BIRCH. The supervised learning models at layer 3 are SVM, RF and NN. The results are summarized in Table 5 and are very similar for all combinations. For the SVM, RF and NN models several combinations of their parameters were tested, and no particular set of values provided results significantly better than the others.

The evaluation results in Table 5 show that the best combination is K-Means for the first and second layers and RF for the third layer. However, it is only slightly better than all the other combinations. Owing to the layered structure and the optimised choice of window size and number of clusters parameters, all the combinations of classification systems offer high values of precision, recall, accuracy and F1 score.

Focusing on the K-Means and RF combination, Table 6 contains, for each user activity, the mean and standard deviations for recall, precision and F1-score obtained for 100 iterations. In addition, a confusion matrix is shown in Fig. 14 to aid in visualising the relationships between traffic activities. The rows of the matrix represent the predicted classification distribution for a single activity, while the columns show the true activity distribution for each predicted class. In the main diagonal of the matrix, the recall for each class is shown.

Results from Table 6 and Fig. 14 show that the precision and recall results are outstanding, with values over 95%, for all except web activities. Web activity provided the worst results because it shares its feature space with all the other activities and presents the behaviour with the greatest variability. It is common to find idle intervals in web browsing and request-response transactions that are like those in interactive traffic. Meanwhile, the Video and Web classes are sometimes mistaken because their activities are sometimes mixed, for example, when using web pages to access the content in video activities and when there are video ads present on web pages. The mean accuracy of the system for K-Means and RF combination is 96.09% (Table 6) showing

Table 5
Experimental scores for different options of classification models used at each layer.

Unsupervised clustering model (layers 1-2)	Supervised classification model (layer 3)	Precision	Recall	F1-Score	Accuracy
K-Means	SVM	0.9350	0.9330	0.9330	0.9574
K-Means	RF	0.9609	0.9541	0.9566	0.9737
K-Means	NN	0.9400	0.9371	0.9374	0.9628
GM	SVM	0.9259	0.9268	0.9234	0.9478
GM	RF	0.9559	0.9551	0.9545	0.9736
GM	NN	0.9404	0.9402	0.9387	0.9604
BIRCH	SVM	0.8937	0.8686	0.8746	0.9098
BIRCH	RF	0.9520	0.9495	0.9497	0.9709
BIRCH	NN	0.9372	0.9259	0.9298	0.9552

Table 6
Experimental results by activity class for the main dataset.

ACTIVITY	Precision Mean	Precision Std	Recall Mean	Recall Std	F1 score Mean	F1 score Std
Bulk	0.991	0.010	0.990	0.008	0.990	0.005
Video	0.958	0.031	0.952	0.020	0.955	0.019
Web	0.875	0.073	0.872	0.064	0.871	0.050
Interactive	0.970	0.019	0.985	0.009	0.977	0.009
Idle	0.991	0.007	0.990	0.006	0.991	0.004

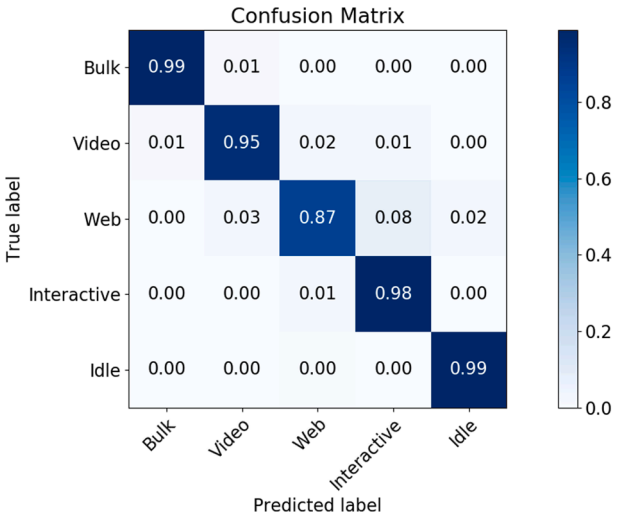

Fig. 14. Confusion matrix for the main dataset.

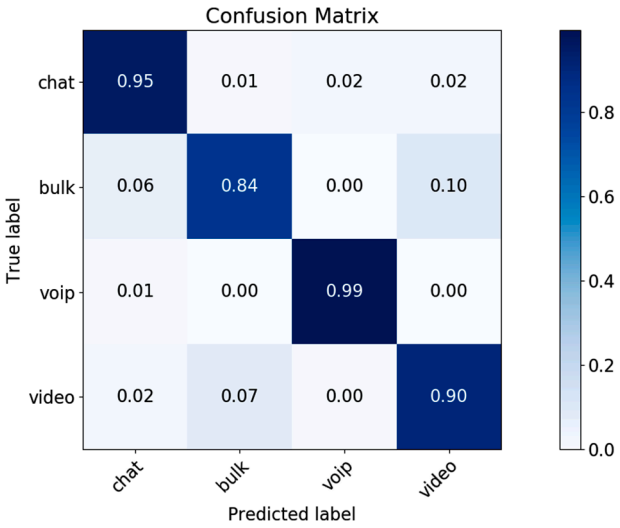

Fig. 15. Confusion matrix for external dataset.

that the CWs are assigned by the system with high accuracy.

We apply our proposal to the external dataset provided by [6] and compare the results to those provided in that paper. This is a dataset that is manually labelled using traffic classes different to our proposal. We have used our three-layered architecture with the new traffic classes and this external dataset, obtaining the results shown in Table 7. The test is performed as proposed in [6], using a 10-fold cross validation and comparing the results using mean values of precision and recall (Table 7). Our proposal improves precision and recall in 3 out of the 4 traffic classes. Checking the confusion matrix in Fig. 15, the problem with this external dataset is the overlap between the video and bulk classes. For both classes the traffic traces have long periods of time without network activity, making it very difficult to distinguish between them. In the architecture presented in this paper, the decision window is a constant CW of 5 s. This short window allows online operation, while in [6] the best decision window is 15 s long, three-times larger and worse for online operation.

6. Related work

Network traffic classification has been a growing research field in recent years and multiple approaches can be found in the literature. Considering the machine learning algorithms used, there are supervised [5,7,10,15] unsupervised [21,2], and hybrid learning approaches [16, 1]. Considering the type of classification, efforts have been made to infer the application network protocol [1,2,7], while other works are aimed at obtaining the application creating the traffic flows [16,10,21] or classifying the traffic by its network behaviour [5,10,15]. Regarding the technique employed, some authors make use of the first n packets of a flow [1,2,7], while other approaches are based on temporal windows [16]; however, all of them allow a near online classification. There are also offline techniques that use the entire captured trace [5,10,15,21]. Additionally, the vast majority of methods perform the classification of the traffic flows independently [1,2,5,7,15,21], while all flows are considered together more infrequently [16].

Table 7
Experimental results by activity class for external dataset.

	Proposal at [6]		Our proposal	
Activity	**Precision**	**Recall**	**Precision**	**Recall**
Video	0.82	0.74	0.97	0.90
Chat	0.83	0.82	0.84	0.95
Bulk	0.83	0.85	0.74	0.84
VoIP	0.98	0.98	0.995	0.995

The authors of [15] used Nearest Neighbour and Linear Discriminant Analysis to classify traffic flows from protocols like HTTPs, FTP, and Telnet into interactive, Bulk, Streaming and Transactional traffic behaviours. The characterisation of the flows was performed using features from full traffic flows at different levels: Packet, Flow, Connection, Intra-flow and Multi-flow. In [5], the traffic flows were classified with a network behaviour-based set of activity classes. The characterisation is made using packet count, size and time-related features from the full traffic flows with an improved SVM algorithm. Deep learning is used in [10] to perform both application identification and traffic characterisation. An automatic deep learning-based feature extraction procedure is proposed with a Neural Network based classifier. The authors of [7] suggest an SVM-based procedure to perform TCP traffic classification that infers the application layer protocol used by each flow, learning from a dataset with HTTP, POP3, FTP and SSH, among others. The flows are characterised using the size and direction of the n (from 3 to 6) first packets of the flow, ignoring packets such as TCP acknowledgements, keepalives, etc., that have an empty payload. K-Means, an unsupervised learning algorithm, is used by the authors of [21] to perform application identification with the captured traffic flows kept separate using a feature selection algorithm. They also compare the results obtained when the selected features are applied directly to the model compared with a previous logarithmic transformation.

In [1], efforts were made to obtain the protocols used in the traffic flows. The authors used a hybrid combination of K-Nearest Neighbour and K-Means to distinguish among protocols like HTTP, SMTP, POP3 or Skype, using features obtained from the number of packets, the size of those packets, the download and upload rates, and the transport protocol (TCP or UDP). In [2], traffic flows were associated with protocols like Edonkey, FTP, HTTPS and SSH using K-Means. Each flow was characterised using only the size of the first five packets of that flow while ignoring TCP control packets like SYN and ACK. Once the clusters were defined, a payload processing tool was used in the training phase to obtain the underlying protocol and allow a cluster-application association that is then used in the online classification. In [6] a two stages scheme is proposed using C4.5 decision tree and K-Nearest Neighbors, but needing to define an adhoc flow timeout depending on the scenario from 15 to 120 s. In [9], the classification is limited to video streaming traffic using Naive Bayes models and consideration of the computational complexity. Our proposal is not limited to video traffic and, computational complexity is taken into consideration while tuning the model.

All these papers, while being substantially different, have some common features that are improved by our proposal. In each earlier study, the classification is made by considering individual traffic flows,

instead of using all the traffic generated by a user. Furthermore, those that allow online classification considered the first n packets, so they are not using all the data present in each flow. Conversely, some techniques classify the flows after the communication has ended, so online classification is not feasible. In our case, all the traffic per flow is used and online classification is possible because of the window-based approach.

A combination of K-Means and SVM is used in [16] to infer the underlying application in temporal windows of mobile network traffic. K-Means is used to obtain clusters from different flows in a shorter temporal window. Those clusters are then grouped and encoded to serve as features for the SVM algorithm. All the clusters obtained from the measurement windows are merged, causing the loss of information on how many flows there are and how they behave individually. Therefore, one of our improvements is the introduction of a flow level window allowing the system to use this information. Furthermore, instead of providing traffic data to the system only until a confident prediction is made, we fixed the length of the CW so continuous monitoring can be performed even in the event of a change in user activity.

In [16], the use of an overly short (5 ms) Behavioural Window forces the system to be trained with specific application patterns instead of network usage behaviour, making the classification of new applications more difficult. Also, our experiments show that the use of a small Behaviour Window has a considerable impact on computational speed and memory usage since the number of samples for the K-Means layer is increased. This leads to longer training and testing time, so the classification of online traffic is compromised when multiple users are being monitored.

Performance of machine learning schemes is important for online classification,. In [11], the authors use a specific structure designed to store parameters of a decision tree to take the smallest memory space to fit it into an FPGA and to reduce the number of memory accesses. The authors in [22] present a classification system for QoS, implemented at kernel level in a FreeBSD firewall to reduce system calls at user level. It can use a decision tree or a Naïve Bayes classifier. In this case, performance is critical because decisions have to be performed for each packet. Also, the complexity of extracted features is important for the performance of the system, as proposed in [18], where features are selected among those less expensive to obtain. It also proposes to have one decision tree for each feature and later joining the results of all trees, improving the final performance specially if implemented in a FPGA. The importance of feature extraction is also described in [8], where processing is made scalable thanks to a computer cluster using Hadoop for a large set of complex features. In our proposal, the features used in the classification are simple, and limited metrics are accumulated in counters for all packets in each window. This allow us to apply feature extraction and processing for online operation.

7. Conclusions

In this paper, we propose a new hybrid system to classify network traffic into a simple set of user activities, based on the network behaviour exhibited by the underlying application. These traffic classes are of use in QoS differentiation and user profiling. Unlike the vast majority of previous studies, the classification is not performed for the different traffic flows individually; instead, all the traffic generated by the user is taken into consideration. Additionally, the three window-based approach uses all packets, unlike the most common techniques based on only the first packets in each flow. The definition of the *Classification Window* (CW) provides the system with enough information to obtain consistent metrics and an online classification.

The use of the *Behaviour Window* (BW) allows us to characterise the flows based on behaviour in a set of small-time intervals. Similarly, the *Flow Window* (FW) permits the differentiation of activities based on the flows present in its network traffic, instead of only the global behaviour of the CW. Up to 97% global accuracy is attained by using the hierarchical multilayer architecture in both the windowed data structure and the classification system. The best combination of models were K-Means for the first two classification layers, and a Random Forest in the last layer.

The computational cost has also been considered to provide the best architecture for online classification of user activities. In addition, the multilayer architecture allows the system to be parallelised and implemented in a distributed architecture.

Future work can approach different scenarios where it is necessary to identify specific applications rather than general traffic classes, for example, security and billing scenarios. Specifically, our future work targets intrusion detection problems and the classification of traffic from benign applications and different types of malware (such as ransomware or malware). The classification of simultaneous activities performed by one user can also be considered, thereby obtaining more than one label for each window.

CRediT authorship contribution statement

Víctor Labayen: Software, Data curation, Investigation, Writing - original draft, Validation. **Eduardo Magaña:** Conceptualization, Methodology, Investigation, Writing - original draft, Supervision. **Daniel Morató:** Conceptualization, Writing - review & editing. **Mikel Izal:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by Spanish MINECO through project PID2019-104451RB-C22.

References

[1] Roni Bar-Yanai, Michael Langberg, David Peleg, Liam Roditty, Realtime classification for encrypted traffic, in: Proceedings of the 9th International Symposium on Experimental Algorithms, SEA, 2010, pp. 373–385, https://doi.org/10.1007/978-3-642-13193-6_32. May 2010.

[2] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, Kave Salamatian, Traffic classification on the fly, Comput. Commun. Rev. 36 (2) (2006) 23–26, https://doi.org/10.1145/1129582.1129589. ISSN 0146-4833.

[3] Yi Zeng, Thomas M. Chen, Classification of traffic flows into QoS classes by unsupervised learning and KNN clustering, KSII Trans. Internet Inf. Syst. 3 (2) (2009) 134–146, https://doi.org/10.3837/tiis.2009.02.001. ISSN 1976-7277.

[4] Víctor Labayen, Eduardo Magaña, Daniel Morató, Mikel Izal, Network traffic and code for machine learning classification, Mendeley Data (2020), https://doi.org/10.17632/5pmnkshffm.2 v2, feb 2020, Download at.

[5] Lei Ding, Fei Yu, Sheng Peng, Chen Xu, A classification algorithm for network traffic based on improved Support Vector Machine, J. Comput. 8 (4) (2013) 1090–1096, https://doi.org/10.4304/jcp.8.4.1090-1096. ISSN 1796-203X.

[6] Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun and Ali A. Ghorbani. Characterization of Encrypted and VPN Traffic Using Time-Related Features. Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), pp. 407-414, Rome, Italy.

[7] Alice Este, Francesco Gringoli, Luca Salgarelli, Support Vector Machines for TCP traffic classification, Comput. Netw. 53 (14) (2009) 2476–2490, https://doi.org/10.1016/j.comnet.2009.05.003. ISSN 1389-1286.

[8] Ahmad M. Karimi, Quamar Niyaz, Weiqing Sun, Ahmad Y. Javaid, Vikay K. Devabhaktuni, Distributed network traffic feature extraction for a real-time IDS, in: Proceedings of the IEEE International Conference on Electro Information Technology (EIT), Grand Forks, ND, 2016, pp. 0522–0526, https://doi.org/10.1109/EIT.2016.7535295, 2016.

[9] Lopes Klenilmar, Almeida Mateus, Matos Walmir, Errico Luciano de, An innovative approach for real-time network traffic classification, Comput. Netw. 158 (2019) 143–157, https://doi.org/10.1016/j.comnet.2019.04.004. ISSN 1389-1286.

[10] Mohammad Lotfollahi, Ramin Shirali, Mahdi Jafari Siavoshani, Mohammadsadegh Saberian, Deep Packet: a novel approach for encrypted traffic classification using deep learning, Soft Comput.. Vol.24 (3) (2020) 1999–2012, https://doi.org/10.1007/s00500-019-04030-2. ISSN 1432-7643.

[11] Yan Luo, Ke Xiang, and Sanping Li. Acceleration of decision tree searching for IP traffic classification. In Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '08).

Association for Computing Machinery, New York, NY, USA, 40-49. DOI 10.1145/1477942.1477949.
[12] Moreno Victor, Ramos Javier, Santiago del Río Pedro, Luis García-Dorado Jose, Jose Gomez-Arribas Francisco, Aracil Javier, Commodity Packet Capture Engines: Tutorial, in: Proceedings of the IEEE Communications Surveys & Tutorials Cookbook and Applicability 17, 2015, pp. 1364–1390, https://doi.org/10.1109/COMST.2015.2424887.
[13] Fabian Pedregosa Scikit-learn, et al., Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830. ISSN 1532-4435.
[14] Paula Roquero, Eduardo Magaña, Rafael Leira, Javier Aracil, Performance evaluation of client-based traffic sniffing for very large populations, Comput. Netw. 166 (2020) 1–10, https://doi.org/10.1016/j.comnet.2019.106985. ISSN 1389-1286.
[15] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, Nick Duffield, Class-of-Service mapping for QoS: a statistical signature-based approach to IP traffic classification, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC 2004), Taormina, Italy, 2004, pp. 135–148, https://doi.org/10.1145/1028788.1028805. Oct.
[16] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic, in: Proceedings of the 10th USENIX Workshop on Offensive Technologies (WOOT 16), 2016, pp. 1–10. Aug.
[17] Scapy, https://scapy.net, Last access: 02-2020.
[18] Da Tong, Yun Rock Qu, K Viktor, Prasanna. accelerating decision tree based traffic classification on FPGA and multicore platforms, IEEE Trans. Parallel Distrib. Syst. 28 (11) (2017) 3046–3059, https://doi.org/10.1109/TPDS.2017.2714661, 1 Nov.
[19] Luis Miguel Torres, Eduardo Magaña, Daniel Morató, Santiago Garcia-Jimenez, Mikel Izal, TBDClust: Time-based density clustering to enable free browsing of sites in pay-per-use mobile internet providers, in: J. Netw. Comput. Appl., December, 99, Elsevier, 2017, pp. 17–27, https://doi.org/10.1016/j.jnca.2017.10.007. ISSN 1084-8045.
[20] Tshark. The Wireshark Network Analyzer 3.0.1, https://www.wireshark.org/docs/man-pages/tshark.html, Last access: 02-2020.
[21] Liu Yingqiu, Li Wei, Li Yunchun, Network Traffic Classification Using K-means Clustering, in: Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007), Iowa, USA, 2007, pp. 360–365, https://doi.org/10.1109/IMSCCS.2007.52. Aug.
[22] Sebastian Zander, Grenville Armitage, Practical machine learning based multimedia traffic classification for distributed QoS management, in: Proceedings of the IEEE 36th Conference on Local Computer Networks, Bonn, 2011, pp. 399–406, https://doi.org/10.1109/LCN.2011.6115322, 2011.

VICTOR LABAYEN graduated on Telecommunication Engineering in 2019 from the Public University of Navarre (UPNA), Spain. During 2018/2019 he held a scholarship on the Electrical, Electronic and Communications Department. Since 2019 he is a network traffic analyst at Naudit High Performance Computing and Networking S.L., a company specialized in network traffic analysis.

EDUARDO MAGAÑA received his M.Sc. and Ph.D. degrees in Telecommunications Engineering from Public University of Navarra, Pamplona, Spain, in 1998 and 2001, respectively. Since 2005, he is associate professor at Public University of Navarra. During 2002 he was a postdoctoral visiting research fellow at the Department of Electrical Engineering and Computer Science, University of California, Berkeley. His main research interests are network monitoring, traffic analysis and performance evaluation of communication networks.

DANIEL MORATO received the M.Sc. degree in Telecommunication Engineering and the Ph.D. degree from the Public University of Navarre, Spain. During 2002 he was a visiting postdoctoral fellow at the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. Since 2006 he has been working at the Department of Automatics and Computing, Public University of Navarre, as an associate professor. His research interests include high-speed networks, performance and traffic analysis of Internet services and network monitoring.

MIKEL IZAL received his M.Sc. and Ph.D. degrees in telecommunication engineering in 1997 and 2002 respectively. In 2003 he worked as a scientific visitant at Institute Eurecom, Sophia- Antipolis, France, performing measures in net- work tomography and peer-to-peer systems. Since then, he has been with the Department of Automatics and Computing of the Public University of Navarre where he is an Associate Professor. His research interests include traffic analysis, network tomography, high speed next generation networks and peer to peer systems.